IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 19 August 2022, accepted 14 September 2022, date of publication 20 September 2022,
date of current version 29 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3208139

RESEARCH ARTICLE

Umformer: A Transformer Dedicated to Univariate Multistep Prediction

MIN LI, QINGHUI CHEN, GANG LI, AND DELONG HAN
Shandong Computer Science Center, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China

Corresponding author: Gang Li (lig@qlu.edu.cn)

This work supported in part by the Shandong Provincial Natural Science Foundation of China under Grant ZR2020MG075, and in part by the Plan of Youth Innovation Team Development of Colleges and Universities in Shandong Province under Grant SD2019-161.

ABSTRACT Univariate multi-step time series forecasting (UMTF) has many applications, such as the forecast of access traffic. The solution to the UMTF problem needs to efficiently capture key information in univariate data and improve the accuracy of multi-step forecasting. The advent of deep learning (DL) enables multi-level, high-performance prediction of complex multivariate inputs, but the solution and research of UMTF problems is extremely scarce. Existing methods cannot satisfy recent univariate forecasting tasks in terms of forecasting accuracy, efficiency, etc. This paper proposes a Transformer-based univariate multi-step forecasting model: Umformer. The contributions include: (1) To maximize the information obtained from a single variable, we propose a Prophet-based method for variable extraction, additionally considering some correlated variables for accurate predictions. (2) Gated linear units variants with three weight matrices (GLUV3) are designed, as a gating to improve the function of selective memory in long sequences, thereby obtaining more helpful information from a limited number of univariate variables and improving prediction accuracy. (3) Shared Double-heads Probsparse Attention (SDHPA) mechanism reduces memory footprint and improves attention-awareness. We combine the latest research results of current DL technology to achieve high-precision prediction in UMTF.Extensive experiments on public datasets from five different domains have shown that five metrics demonstrate that the Umformer approach is significantly better than existing methods. We offer a more efficient solution for UMTF.

INDEX TERMS Multi-step, univariate, time series forecasting, transformers.

I. INTRODUCTION

Time series data (TSD) is applied widely in agriculture, business, meteorology, and many other fields [1], [2]. Researchers were asked to focus on discovering internal patterns in TSD and making predictions [3]. Traditional univariate prediction methods usually use the TSD from the previous steps as model input, the data from the next step as labels. Due to the minimal number of features in the univariate data, traditional models for solving the UMTF problem learn very little information, resulting in inaccurate results. Moreover, some other challenges have remained. For example, applying the latest multivariate time series forecasting [4], [5], [6] to specific univariate forecasting is hard, including extracting features and improving accuracy.

UMTF currently faces two key challenges: one is to be able to effectively extract some of the features that influence prediction from the limited data available. The second is to improve the multi-step prediction accuracy of the model to meet the current and growing needs of society. These challenges require researchers to provide:

1) a mechanism for univariate variable feature extraction scientifically extracting feature variables in TSD;

2) a DL model dedicated to UMTF, which has efficient computing power.

Recent studies have found that the Prophet algorithm [7] can effectively extract time variables, and the TFT model [8] can accurately predict multivariate data in multiple steps. The Prophet algorithm is based on the time series decomposition

The associate editor coordinating the review of this manuscript and approving it for publication was Sajid Ali.

technology, which decomposes each data in a TSD into four parts, i.e., seasonal, trend, holiday, and remaining. Its data decomposition method excels in univariate forecasting and is currently one of the most popular tools for data analysts.

The TFT model introduces a variety of heterogeneous techniques into the structure of the Transformer [9]. It provides inputs of known variables for prediction in the future and improves the training capability by performing feature selection and weight calculation of different variables. TFT is a significant innovation in time series forecasting. Although TFT has emerged as a sophisticated method for solving multivariate prediction problems, it cannot solve the UMTF problem. The reasons include:

1) Univariate data limits the learning capability and cannot provide the variety of variables necessary for the TFT.

2) The Gated Linear Unit (GLU) [10] module in TFT mimics and improves upon the LSTM [11] in terms of selectively remembering and acquiring information about long sequences. However, when the feature variables are limited, the GLU is less effective in accurately extracting the critical information for the UMTF.

3) The Interpretable Multi-Head Attention mechanism used by TFT has high time complexity. It is also unable to capture long-term dependent focus when making univariate predictions.

This paper proposes a Transformer-based univariate multistep prediction framework: Umformer. Based on the above two models, the method achieves high performance prediction for univariate data. Contributions are as follows.

1) This paper proposes a data processing method based on the Prophet algorithm, followed by setting variables at specific time points. We considered the impact of multiple real-world variables on the forecast, combining the specificity of UMTF data and specific problems, the added feature variables are classified into Static seasonal variables, Past observed variables, and Known time variables, which can effectively overcome the feature limitations of TSD.

2) GLUV3 is defined to replace the original gated linear unit. The new variant is more conducive to meeting the denoising targets of the pre-training phase and improves the model's accuracy.

3) The SDHPA mechanism is presented to ensure low time complexity and improve the attention-aware capability and prediction accuracy.

4) On various real data sets we show how umformer can be applied in practice, as well as comparing existing methods and demonstrating the advantages of the proposed approach.

II. RELATED WORK

A. UNIVARIATE TIME SERIES FORECASTING

Time series forecasting methods are categorized into univariate and multivariate forecasting. The methods [12] are used for univariate forecasting [13], [14], [15] based on the observation data of a given sequence. The multivariate forecasting methods [16], [17] combine the observation data of exogenous variables for prediction [18]. This paper mainly studies the methods of univariate sequence prediction.

The Autoregression (AR) [19] is a linear regression model that describes random variables at a future time in terms of a linear combination of random variables at a particular time in the previous period, which was the standard method used in early time series forecasting [20], [21]. However, it has high requirements for data autocorrelation and can only be used to predict scenarios that are heavily influenced by historical factors, but not those that are heavily influenced by social, natural, and other factors. Moving average(MA) [22], [23] uses a moving average of white noise to simulate TSD and calculates the average of the historical data as the forecast for the next period. When there are more forecast data, a large amount of data needs to be stored. Many studies have shown that the forecast accuracy of moving averages is low [24]. Autoregressive Moving Average (ARMA) [25], [26] combine AR and MA models to reduce the number of past parameters, Autoregressive Integrated Moving Average Model(ARIMA) [27], [28], [29], [30] is a model built by regressing the lagged values of the dependent variable on the present and lagged values of the random error term, both of them require the TSD to be stable after differentiation and can only capture linear relationships. Exponential smoothing (ES) [31], [32], [33] is a forecasting method that introduces the smoothing factor, a simplified weighting factor, to obtain a time series of averages based on the actual quantity and the forecast quantity for the current period of a particular indicator. It is a particular weighted average method in which historical data closer to the forecast period is given with a larger weight, and the weight decreases exponentially. But it cannot discriminate the data turning point and is mainly used for short-term forecasting.

In recent years, models based on deep learning have been used for time series forecasting, including convolutional neural networks(CNNs) and recurrent neural networks(RNNs). The LSTM [34], [35], a kind of special RNN, is currently used in practical prediction applications for the future by selectively memorizing sequences [36]. However, one of the main limitations of using LSTM to predict time series is that the model relies heavily on asymptotic forecasting, so remote forecasting may not be effective. Moreover, it is highly prone to hysteresis [37]. In recent years, the Transformer [38], [39] has gradually been used for time series prediction [40]. The seq2seq [41] model based on the Attention mechanism has emerged in prediction [42], but it has not yet been used in univariate time series prediction. A study [43] comparing Transformer and LSTM solutions to prediction problems pointed out the limitations of Transformer in terms of computation and parameter handling.

B. MULTI-STEP TIME SERIES FORECASTING

Based on the above discussion, there is another vital issue: an inaccurate grasp of data characteristics affects the accuracy of multi-step time series prediction. TBPTT [44] does not modify the input when calculating the gradient and only uses

the gradient from the previous step to estimate the current weight update. The literature [45] used the auxiliary loss to learn the dependencies in the sequence and add auxiliary to strengthen the gradient flow. But its performance is still not optimistic. Other attempts include explaining the breadth of Graph neural networks [46] to learn long-term dependencies. These methods try to improve the long-path gradient flow of the recursive network, but in the UMTF problem, the performance is limited due to the increase of the predicted time step. CNN-based methods [47], [48], [49] use convolutional filters to capture long-term dependence, and their receptive fields increase exponentially with the number of layers, which impairs the alignment between sequences. In the multi-step accurate prediction, the main task is to extract the required data from multiple input time steps, which requires more output. Therefore, the above methods cannot be directly used for univariate multi-step forecasting. Attention-based models proposed addictive attention [50] to improve the word alignment of the encoder-decoder structure in translation tasks. The self-attention-based Transformer [9] has recently been proposed as a new idea of sequence modeling and has achieved great success, especially in NLP. Various studies have proved that it has better sequence tracking capabilities.

In order to reduce the time complexity, many researchers presented heuristic methods to reduce the complexity of the self-attention mechanism to O(LlogL) [51], but their efficiency is limited. Reformer [52] achieves O(LlogL) through locally sensitive hash self-attention, but its application scope has been limited. Using the hidden auxiliary state to capture remote dependencies [53] may increase the time complexity, but it may not necessarily improve efficiency. The ProbSparse self-attention [54] reduces the time complexity and space complexity. It maintains the original accuracy but does not consider interpretability. In the Interpretable Multi-Head Attention of the TFT [8], multi-head attention shares the value of each head, and aggregates all the heads to enhance the interpretability. However, it does not consider accuracy and efficiency when self-attention is used.

In our work, a Transformer-based Umformer is proposed for multi-step accurate prediction. The most relevant works [55], [56] all use Transformer in TSD, but as they use an ordinary Transformer, multi-step prediction is not very effective. In addition, other works [27], [52], [57] have noted the scarcity of self-attention mechanisms and discussed them in the main context of long-term dependencies based on Transformers, such as a sparse cluster-based Transformer for remote dependency encoding [40].

III. PREDEFINE

Our research aims to provide a solution to the UMTF problem by addressing the problem of temporal feature decomposition in this domain and improving the accuracy of multi-step prediction.

To begin with, we need to define the UMTF problem. We provide the machine input: $I = \{x_{t-n+1}, \ldots, x_{t-2}, x_{t-1}, x_t\}$, where each x is univariate data observed before the current timestamp t. Then the predicted output values are output by the prediction model: $Y = \{y_{t+1}, y_{t+2}, \ldots, y_{t+m}\}$, where each y in the model output Y is the value of the data for an equal time difference after the current timestamp t.We assume that n observations are input to the model, predicting values for m future time steps. Specifically, we input n timestamps and corresponding label values up to the current timestamp t into the model, and the model predicts and outputs the values at m time steps in the future to complete the prediction.

IV. METHODOLOGY

The initial methods used for time series forecasting were mainly mathematical and statistical models, including AR [58], MA [59], ARMA [60], and ARIMA [28]. In recent years, recurrent neural networks (RNNs) [35] and convolutional neural networks (CNNs) [61] have also been gradually used in prediction. The Attention mechanism-based seq2seq model [62] and the Transformer lead to a new round of research on time series forecasting.

This paper is dedicated to addressing the challenges of current univariate prediction methods and finding optimal solutions. We propose an Umformer framework(Future 1) that is unique and novel in terms of feature engineering, model construction, etc. Major components of Umformer are

(1) Univariate TSD feature engineering: This includes prophet-based feature decomposition, data pre-processing, data classification, and feature selection. The consideration of multiple variables greatly improves the effectiveness of univariate prediction.

(2) Sequence to sequence: The encoder and decoder are used for the input and output of the time series respectively, predicting the change in TSD based on the first few inputs.

(3) GLUV3: as a gating to improve the function of selective memory in long sequences, thereby obtaining more helpful information from a limited number of variables.

(4) Transformer decoder: The designed Static Variable Enhancement GRN (SVEGRN) considers the effect of seasonal variables, and the proposed SDHPA reduces memory footprint and improves attention-awareness, increasing prediction accuracy.

Figure 1 demonstrates the entire process of univariate multi-step prediction by the Umformer model. First, univariate data with feature expansion is classified. After the feature selection, the static seasonal variables enter the separate GRN encoders. Different vectors are generated and enter the seq2seq model. the Static Variable Enhancement GRN(SVEGRN) enhances the temporal characteristics of static seasonal variables. The Known time variables enter the encoder, and the known dynamic variables are used for feature selection. The output from SVEGRN enters the SDHPA after the GLUV3 and GRN layers. In this process, Static seasonal variables affect the calculation of GRN again and enhance the static characteristics. After the feature variable selection and the LSTM-based encoder/decoder, the information is screened through a layer of GRN. Finally, the results of

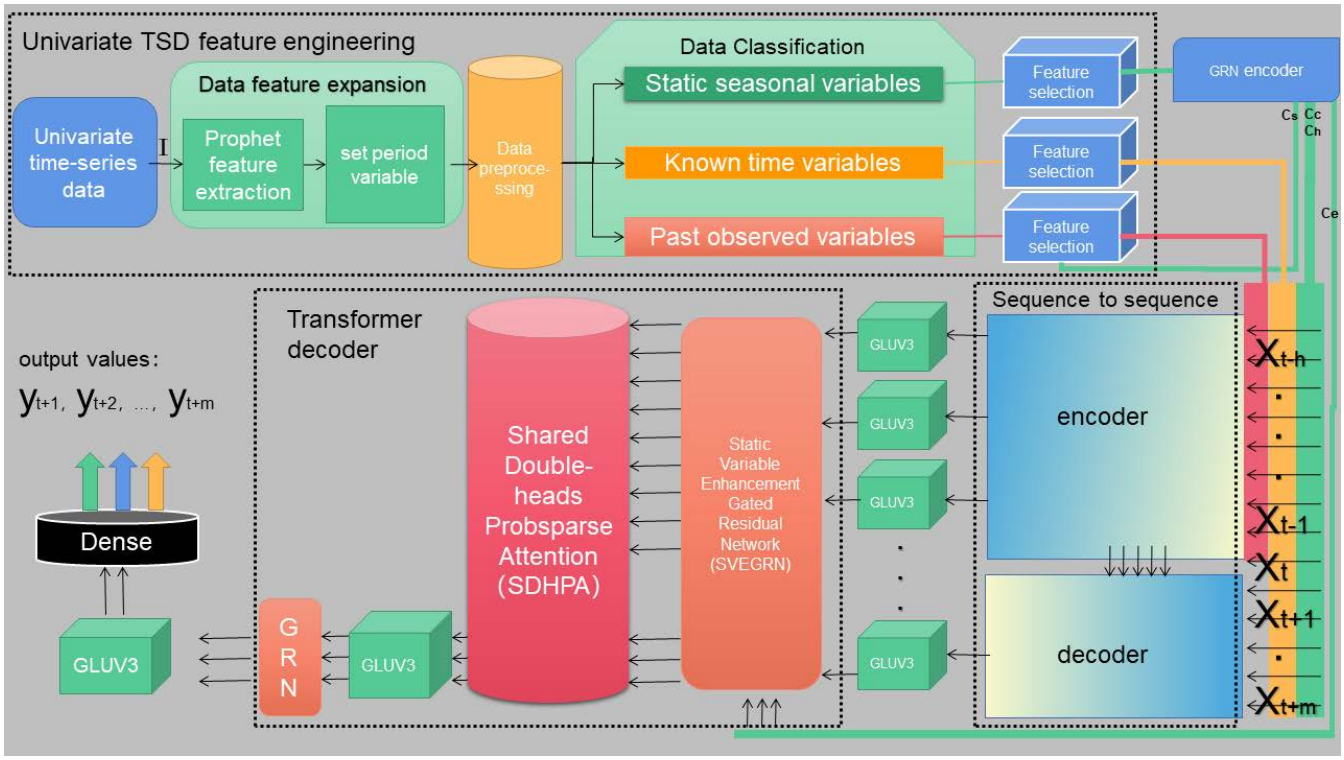


FIGURE 1. Umformer framework.

the Attention mechanism are input into GLUV3, GRN, and GLUV3 in turn, and the predicted value is output through a dense layer.

A. UNIVARIATE TSD FEATURE ENGINEERING

1) FEATURE DECOMPOSITION AND DATASET PREPROCESSING

In this stage, multiple data features are extracted and expanded by Prophet algorithm from the original data. Suppose $y(t)$ is a time series, $s(t)$ is the periodicity in weeks or years, $g(t)$ is non-periodical change trend of $y(t)$, $h(t)$ is whether there are holidays in the day, and ε_t is the error term.

$$y(t) = s(t) + g(t) + h(t) + \varepsilon_t \tag{1}$$

Umformer selectively extracts characteristic variables from the feature extraction of Prophet. We also added time terms such as year, month, week, and day to the original variable.

In addition, we use research methods such as scientific reasoning and practical analysis to add fundamental variables that affect the prediction outcomes of specific data sets, such as bool-type holidays and major events. For univariate prediction, we combine deep learning and mathematical statistics techniques to achieve data feature extraction and scaling.

The Prophet is based on time series decomposition and machine learning for univariate time series forecasting. Its internally improved time series decomposition technology provides us with new solutions. The time series $y(t)$ are decomposed into the following parts, The period term $s(t)$ represents the periodicity in weeks or years; the trend term $g(t)$ represents the non-periodical change trend of the time series; the holiday term $h(t)$ represents whether there are holidays in the day; the remaining term ε_t represents the error term or is called the residual term, when $y(t) = s(t) + g(t) + h(t) + \varepsilon_t$, the Prophet algorithm is to fit these items, and then finally add them to get the predicted value of the time series. We use the variable processing method of the algorithm to apply to data preprocessing and decompose the data to obtain multivariate time series data.

In conclusion, we have solved one of the bottlenecks of univariate time series forecasting - univariate TSD cannot be predicted with existing multivariate time series forecasting models, it has too few correlated variables. We extracted some important relevant variables through the prophet feature extraction method, and added periodic variables, which will improve the predictive ability of the model.

2) DATA CHARACTERISTICS CLASSIFICATION

This paper divides the data into Static seasonal variables, Past observed variables, and Known time variables.

1) Static seasonal variables are seasonally relevant characteristics. As seasonal variables remain constant over a given continuous-time data, they can be used as static covariates to control the overall situation. The model will pay more attention to changes in data characteristics within the same season,

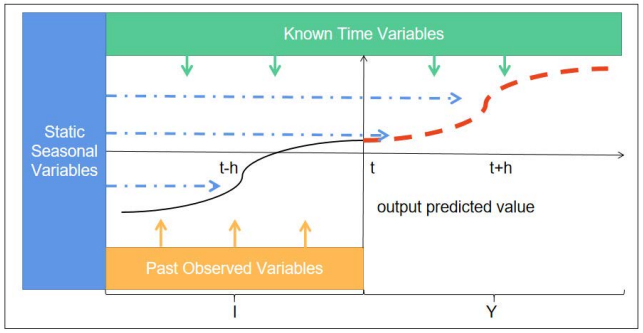


FIGURE 2. Model input and output.

reduce the prediction error caused by seasonal changes and improve the model's ability to fit cyclical factors.

2) Past observed variables are known and observed as the dynamic variables before the prediction point. In this part, the variables are impossible to know when predicting the future time. Past observed variables are essential features that affect the model's prediction, so it is necessary to input the dynamic variables from the previous period to predict the label value for a period later.

3) Known time variables are known in advance throughout the forecasting system. Such variables can influence the training of the model parameters prior to forecasting. They can provide the model with data on variables at various future time points, such as annual, monthly, and daily variables known at the various times around the forecast time t.

Figure 2 shows more clearly how the classified data play a role in the novel model and the Umformer model on how to learn these variables after expanding and classifying the data. First, variables and labels developed by the Prophet algorithm are used as the Past observation variables and become the historical data affecting the future forecasting. The static seasonal variable is the time points, including spring, summer, autumn, and winter. Since seasonal terms do not change over a continuous period and the volume of data is large, they can be used as a global coordination feature. Known time variables can be used in the past or future to influence forecasting.

3) FEATURE SELECTION NETWORK

Past observed variables p, Static seasonal variables s, and Known time variables k are pre-processed with univariate data from the Umformer model and then efficiently extract information through a feature selection network consisting of a series of gated residual networks (GRN). The GRN is used as a component to allow the model to be flexible for non-linear processing. The input to the GRN is defined to consist of two parts, one is the main input i, and the other is the auxiliary input c (the data after the static covariate passes through the encoder) and yields,

$$GRN_{\omega}(i,c) = \text{LayerNorm}\left(i + GLUV3_{\omega}(\eta_1)\right) \quad (2)$$

$$\eta_1 = W_{1,\omega}\eta_2 + b_{2,\omega} \quad (3)$$

$$\eta_2 = \text{ELU}\left(W_{2,\omega}i + W_{3,\omega}c + b_{2,\omega}\right) \quad (4)$$

where ELU is the exponential linear unit(ELU) activation function, $\eta_1, \eta_2 \in R^{d_{model}}$ are intermediate layers, LayerNorm is standard layer normalization, and ω is an index to denote weight sharing. ELU activation will act as an identity function when $W_{2,\omega}i + W_{3,\omega}c + b_{2,\omega} \gg 0$, and will produce a constant output when $W_{2,\omega}i + W_{3,\omega}c + b_{2,\omega} \ll 0$, resulting in linear layer behaviour. We present a GLUV3-based component gating layer to suppress the flexibility of any part of the structure not necessary for a given dataset. Thus GRN can play the role of variable feature selection.

Umformer passes the Past observation variables p, Static seasonal variable s and Known time variables k after classification independently through the feature selection network. Let $p_T^{(j)} \in R^{d_{model}}$ denotes the transformed input of the j-th variable at moment T, and $P_T = p_T^{(1)^T}, \ldots, p_T^{(j_{max})^T}{}^T$ being the flattened vector of all past observation inputs at time T. Feeding both P_T and static covariate c_s to GRN, followed by a Softmax layer, variable selection weights ω_{vs} is obtained that

$$\omega_{vs_T} = \text{Softmax}\left(\text{GRN}_{\text{vs}}\left(P_T, c_s\right)\right) \quad (5)$$

At each time step, an additional non-linear processing layer is added, feeding $p_T^{(j)}$ into its GRN that

$$\tilde{p}_T^{(j)} = \text{GRN}_{\tilde{p}(j)}\left(p_T^{(j)}\right) \quad (6)$$

where $\tilde{p}_T^{(j)}$ is the processed feature vector for variable j. Each variable has its calculation, sharing weights at all time steps T. The processed features are weighted and combined according to their variable selection weights as follows:

$$\tilde{p}_T = \sum_{j=1}^{j_{max}\sum_T \tilde{}^{(j)}} \omega_{vs_T}^{(j)} \quad (7)$$

where $\omega_{vs_T}^{(j)}$ is the j-th element of vector ω_{vs_T}.

The strengths of TFT are mainly reflected in the feature selection of GRN, similar to principal component analysis (PCA) and the explainable multi-head self-attention mechanism. Moreover, its GRN, as a threshold device in TFT, is more like a replacement for the Dense layer. However, compared with the Dense layer, it extracts the effective components and improves the performance and learning efficiency of the model.

The TFT framework is shown in Figure 3. Different types of variables are fed into the corresponding variable selection network. After the sequence-to-sequence model, the multi-head self-attention mechanism gets the weight of each variable into multiple gate and GRN layers, and finally into a multi-step time series prediction value.

Next, in order to make accurate predictions for specific data after univariate feature decomposition, this work improves the GLU layer and the Multi-headed Attention mechanism to increase the accuracy of multi-step prediction. We propose GLUV3 to replace the original GLU to improve

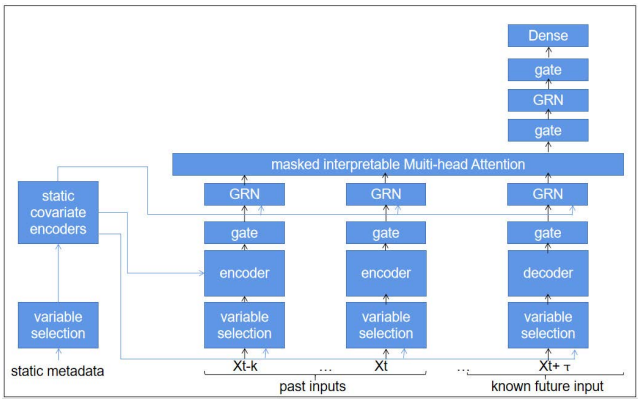


FIGURE 3. TFT model.

the model's selective memory and forgetting functions in long sequences. Moreover, the SDHPA mechanism improves the attention-aware ability of the Attention mechanism under the premise of ensuring time complexity.

B. SEQUENCE TO SEQUENCE

Umformer integrates seasonal information for seasonal variables using a separate GRN encoder, which generates four different vectors c_s, c_e, c_c, c_h,. these vectors are sent to different locations so that the four vectors play different roles in different layers. Specifically, this includes c_s that influences the choice of variables for each variable, c_c, c_h for local processing of temporal features and c_e used in SVEGRN for seasonal information enhancement. For example, ζ is used as the output of the variable selection network and the c vector is encoded by $GRN\ (\zeta)$ for output. Except for the static seasonal variables, the other variables are generated by the LSTM encoder and decoder to produce vectors into the next layer.

In TSD, the label values of each timestamp are usually related to the values around them. However, as past and future inputs are different, this prevents them from being entered together as the same characteristic variable. Therefore, our work applied a seq2seq model to handle these differences with $X_{t-h} : X_t$ going into the encoder and $X_{t+1} : X_{t+m}$ into the decoder. A consistent set of temporal features is generated, which are used as inputs to the next layer. Considering $T \in \{t-h : t+m\}$, t is the current time point, h is the maximum time step known before t, and m is the maximum time step predicted after t.this can also be used as an alternative to the standard position coding to provide an appropriate inductive bias for the time sequence of the input. $\varphi_T \in \varphi(t-h), \ldots, \varphi(t), \ldots, \varphi(t+m)$, where φ_T is the output of encoder and decoder at a certain point in time. In addition, in order to allow seasonal variable data to affect processing, we use c_c, c_h vectors from the above GRN encoder to initialize the cell state and hidden state of the first LSTM in the layer. It also uses gated skip connections after this layer

$$\phi_T = \text{LayerNorm}\left(X_T + GLUV3_\phi\left(\varphi_T\right)\right) \tag{8}$$

C. GLUV3

GLU was initially proposed in language model [10]. Its availability keeps information strictly according to the time sequence position when performing sequential data processing, improving performance and speeding up the operation through the parallel processing structure. GLU was initially defined as the component product of two linear input transformations, and one is activated by the Sigmoid function. The descriptor ignores the activation and calls it the "bilinear" layer [63], which has been explained in a study. Recent studies have proved that GLU can also be changed by using a different activation function instead of the Sigmoid function [64].

This paper proposed GLUV3 to improve the ability to retain important information in chronological order when processing data, and the capacity to selectively forget and remember information is also enhanced.

$$\text{GLUV3}\ (a) = W_1\left[(a * W_2 + \alpha) \otimes \text{Gelu}\ (a * W_3 + \beta)\right] + \chi \tag{9}$$

where a indicates the output of the previous layer and the input of this layer,W_1, W_2 and W_3 are the convolutions kernel parameter, α, β and χ are the bias parameters, Gelu is the Gelu activation function [65]. We replace the Sigmoid activation with a Gelu activation and add a one-dimensional product calculation.

To solve the gradient disappearance of the Sigmoid function, we selected the Gelu function. As an activation function that adjusts the output through a gating mechanism, the idea of random regularity of the Gelu can more conveniently improve the speed of gradient descent and learning. No matter how large the input value is, its derivative will not tend to be 0. To a certain extent, it avoids the problem of gradient disappearance, and its fitting ability is faster and better than Sigmoid.

Figure 4 shows the specific structure of GLUV3. The input of this layer is a series of continuous TSD. A vector represents the original series data. The calculation of the hidden layer is calculated according to the above formula.

The output of each layer has a linear projection $a * W_2 + \alpha$ modulated by the gated Gelu $a * W_3 + \beta$. Similar to LSTM, these gates multiply each element of the matrix. In addition, we add an overall linear projection, so these layers are passed through three weight matrices to improve the accuracy of the calculation.

Our method wraps the convolution and GLU in a pre-activated residual block and adds the input of the residual block to the output. One of the most effective choices is using the Gelu layer and adding a one-dimensional weight matrix, which does training and testing faster and more accurate.

D. SDHPA

This paper has employed a self-attentive mechanism to learn long-term relationships between different time steps, and modified the transformer-based architecture. We propose

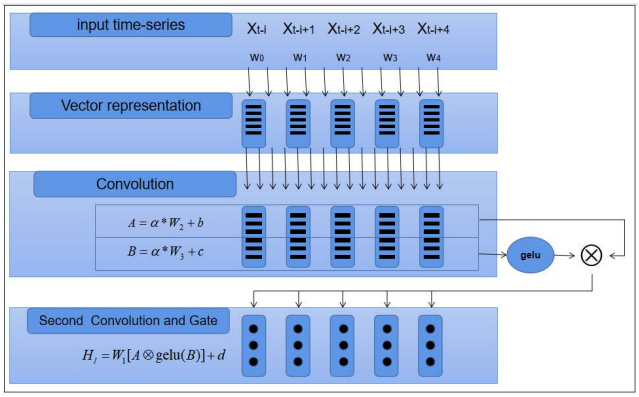


FIGURE 4. GLUV3.

SDHPA specifically applied to the UMTF problem, using the probsparse self-attention mechanism and the double-heads Attention mechanism when performing the feature weight calculation for the Attention mechanism. The Probsparse self-attention mechanism was proposed in the study of Informer [54]. Due to the traditional Scaled Dot-Product self-attention mechanism resulting in a time complexity and memory usage of $O\left(L^2\right)$ per layer, The PorbSpare self-attention mechanism to efficiently replace the canonical self-attention mechanism achieves a time complexity and memory consumption of $O\left(L\log L\right)$. We experimentally demonstrate that this approach gives more accurate predictions than Scaled Dot-Product Attention for the UMTF problem, even though it reduces time complexity and memory usage. To improve interpretation and speed up operations, we summed and averaged each head in the calculation of the Double-heads attention. Due to the limitations of univariate TSD, there is relatively little information represented. When there are too many heads, this can lead to a reduced impact of the representation subspace in each head, not capturing important representations effectively and even leading to overfitting, so we choose a two-headed attention mechanism to pay more attention to important representations.

General, the Attention mechanism is based on the relationship between the key $K \in R^{N\times d_{attn}}$ and the query $Q \in R^{N\times d_{attn}}$, extended for the value $V \in R^{N\times d_{attn}}$. $Attention\left(Q, K, V\right)$ is generally calculated from the traditional scaled dot product attention [9].

The proposal SDHPA is presented as follows. For self-attention, considering random sampling, randomly select $u-th\ K$, get k_u, and evaluate M for $q_i \in Q$ with respect to k_u that

$$\bar{M}\left(q_i, K_u\right) = \max\left(\frac{q_i k_j{}^T}{\sqrt{d}}\right) - \frac{1}{L_K}\sum_{j=1}^{L_K}\frac{q_i k_j{}^T}{\sqrt{d}} \tag{10}$$

where $k_j \in K_u$, L_k is the length of the sampled K_u. Find the largest u number of Q_i in M, form Q_u and calculate $Attention\left(Q, K, V\right)$ according to K_u:

$$\text{Attention}\left(Q, K, V\right) = \text{softmax}\left(\frac{\bar{Q}K^T}{\sqrt{d}}\right)V \tag{11}$$

where $\bar{Q}$ is a sparse matrix of the same size as Q, containing only u queries under the sparse metric $\bar{M}\left(q_i, K_u\right)$.$Q_u$ is the matrix composed of the selected u number of q_i, and the unselected q_i is initialized to the original Q_r matrix by finding the mean value after $Attention\left(Q, K, V\right)$, and the non-zero values in the Q_u matrix are updated to the Q_r matrix to obtain the final Q matrix. In fact, in the self-attentive computation, the input lengths of queries and keys are usually equal, i.e. $L_Q = L_K = L_V$, making the total time complexity and space complexity of ProbSparse self-attentive $O\left(L\ln L\right)$.

Multi-headed attention is used to enable models to jointly focus on information from different representation subspaces at different locations, which is extremely important in the NLP domain for semantic extraction and can also be effective in the UMTF domain when adequately utilized, i.e.

$$\text{MultiHead}\left(Q, K, V\right) = \left[H_1, \ldots, H_{mH}\right]W_H \tag{12}$$

and

$$H_h = \text{Attention}\left(QW_Q{}^{(h)}, KW_K{}^{(h)}, VW_V{}^{(h)}\right) \tag{13}$$

where $W_K^h \in R^{d_{model}\times d_{attn}}$, $W_Q^h \in R^{d_{model}\times d_{attn}}$, $W_V^h \in R^{d_{model}\times d_{attn}}$ are head-specific weights for key, querie and value, and $W_H \in R^{(m_H\cdot d_V)\times d_{model}}$, combine the outputs of all heads H_h. As mentioned above, because each head uses a different value, the weight of attention alone does not guarantee that the importance of a particular feature is reflected and exploited. Therefore, the interpretability is enhanced by modifying multi-head attention to shared values in each head and additive aggregation of all heads when seeking a two-head Attention mechanism.

The formula for finding double-headed attention is as follows:

$$\text{Doublehead}(Q, K, V) = HW_H \tag{14}$$

$$= \frac{1}{2}\left[\text{Attention}\left(QW_Q{}^1, K\ W_K{}^1, V\ W_V{}^1\right) + \text{Attention}\left(QW_Q{}^2, K\ W_K{}^2, V\ W_V{}^2\right)\right] \tag{15}$$

where $W_V \in R^{d_{model}\times d_v}$ are value weights shared across heads, and $W_H \in R^{d_{attn}\times d_{model}}$ are used for final linear mapping. It can be found that different temporal patterns can be learned between the two heads while noticing a common set of input features, which can be interpreted as a simple aggregation of attention.

The final value $Doublehead\left(Q, K, V\right)$, output by the Attention mechanism, goes to the next level of computation.

E. TRANSFORMER DECODER

In order to be able to consider the effect of seasonal variables on forecasts globally, the SVEGRN we built integrates c_e variables and sequence to sequence outputs to improve the efficiency of the model's fit to the variables.

$$\mathrm{SVNGRN}_T = GRN\left(\phi_T, c_e\right) \tag{16}$$

After making the seasonal information augmented, we applies self-attention. All input temporal features are first grouped into single matrix $\vartheta\left(t\right) = [\theta_{t-k}, \ldots, \theta_{t+m}]$, when θ_T is time feature, $T \in (t-h, \ldots, t+m)$, Double-headed interpretable Attention mechanism applied to each prediction time.

$$B\left(t\right) = \mathrm{DoubleHead}\left(\vartheta\left(t\right), \vartheta\left(t\right), \vartheta\left(t\right)\right) \tag{17}$$

Denote $B\left(t\right) = [\beta_{t-h}, \ldots, \beta_{t+m}]$, decoder masking is applied to the multi-headed attention layer to ensure that each temporal dimension can only notice the previous features. After the self-attention layer, an additional gating layer is applied to facilitate training that

$$\gamma_T = \mathrm{LayerNorm}\left(\mathrm{SVEGRN}_T + \mathrm{GLUV3}_\gamma\left(\gamma_T\right)\right) \tag{18}$$

Our approach applies additional non-linear processing and GRN operations to the output of the self-attention layer, and we also apply a gated residual connection that skipped the entire transformer module, providing a direct path to the seq2seq layer.

$$\psi_T = \mathrm{LayerNorm}\left(\phi_T + GLUV3_\psi\left(\psi_T\right)\right) \tag{19}$$

F. OUTPUT AND LOSS

The output of Umformer uses quantile prediction, which is realized by three percentiles, 0.1, 0.5, and 0.9, where $Y_T = y\left(q, T\right) = W_q \cdot \psi_T + b_q$, b_q is the linear coefficient of the specified quantile q. The T is different from the previous one in that ranges only between $\{t+1 : t+m\}$, because the forecast must be at a point in time after t. For different datasets, we find that the prediction levels are different for different quartiles, and we choose the quartile with the best result as the final prediction output.

The quantile loss is defined as

$$E_i = \mathrm{tar} - \mathrm{pred}_i \tag{20}$$

$$L_i = \max(\left(q_i - 1\right) \cdot E, q_i \cdot E) \tag{21}$$

$$\mathrm{loss} = \frac{\sum_{i=1}^{n} L_i}{n} \tag{22}$$

where *tar* is the label value, i takes the value 0,1,2, $pred_i$ are the model prediction output value of the i-th q, and q takes the value 0.1,0.5,0.9.

V. EXPERIMENT

A. DATASETS

We conduct experiments on datasets from three different domains. These data consist of only two columns, including each timestamp and the corresponding label value, and are univariate TSD. The next task is to conduct comparative experiments on our method in these univariate datasets.

TABLE 1. Experimental parameters.

Parameters	Value
ratio of partition(train: validate: test)	6:2:2
Batch size	32
Dropout rate	0.5
Learning rate	0.0001
LSTM layersrate	2
Quantilesrate	0.1,0.5,0.9

1) VISITS

The time series data of daily visits to Peyton Manning's Wikipedia homepage (2007/12/10-2016/01/20). Each variable in the original data for this dataset is a logarithmic (log) value of specific access data.

2) TEMPERATURE

This dataset is called the "Daily Minimum Temperature Dataset", which describes the minimum daily temperature in Melbourne, Australia, for ten years (1981-1990). The unit is Celsius, and there are 3650 observations in total from Australia's Bureau of Meteorology.

3) SUNSPOT

This dataset is called the "Monthly Sunspot Dataset", which describes the monthly count of the number of sunspots observed in the past 230 years (1749-1983). A total of 2,820 observations were counted in the unit.

4) ECN:(A EUROPEAN CITY CORE NETWORK)

Data (in bits) from a private ISP in 11 cities in Europe. The data correspond to transatlantic links and were collected between 7 June 2005, 06:57 and 31 July 2005, 11:17. Data were collected at 5-minute intervals. There are 14,772 data items.

5) TAXI

New York City taxi demand dataset, recording the number of taxis in demand in the city at different times, covering the period 2014-07-01 to 2015-01-32. data was collected at 30 minute intervals. A total of 10,321 data items are available.

B. ENVIRONMENT

The pytorch framework version 1.9.0, combined with CUDA version 11.1 and cuDNN version 8.0, can be used to run the code for this article. The model training and inference in this paper are performed on NVIDIA RTX 3080 and intel i9-9900k@5GHz. NVIDIA RTX 3080 has 4352 CUDA cores and 10GB of GDDR6X video memory, 320 bit width, 760GB/s maximum bandwidth, and 320W TGP. The Intel i9-9900k has eight cores and sixteen threads. Umformer are trained on CPUs in all datasets and can be deployed with

TABLE 2. Components level comparison table.

Components		SDA+GLU				SDA+GLUV2			
Metric		MSE	RMSE	MAE	MAPE	MSE	RMSE	MAE	MAPE
learning cycle	20	0.166	0.408	0.334	4.201	**0.135**	**0.368**	**0.274**	**3.485**
	30	0.180	0.424	0.355	4.451	**0.110**	**0.332**	**0.263**	**3.312**
	40	0.201	0.448	0.369	4.629	0.142	0.377	0.286	3.635
	50	0.222	0.471	0.389	4.903	0.132	0.363	0.281	3.564
	60	0.143	0.379	0.312	3.920	0.103	0.321	0.254	3.210
	70	0.281	0.531	0.440	5.523	0.133	0.364	0.275	3.506
	80	0.163	0.403	0.322	4.066	0.147	0.384	0.292	3.726
Components		SDA+GLU3				SDPA+GLU			
Metric		MSE	RMSE	MAE	MAPE	MSE	RMSE	MAE	MAPE
learning cycle	20	0.136	0.368	0.288	3.681	0.232	0.482	0.398	4.909
	30	0.118	0.345	0.269	3.419	0.214	0.463	0.362	4.581
	40	0.166	0.407	0.323	4.101	0.189	0.435	0.337	4.264
	50	0.204	0.452	0.365	4.623	0.200	0.448	0.359	4.490
	60	0.297	0.545	0.455	5.727	0.197	0.444	0.353	4.432
	70	0.304	0.552	0.455	5.711	0.166	0.408	0.311	3.928
	80	0.150	0.387	0.299	3.814	0.182	0.426	0.329	4.139
Components		SDPA+GLUV2				SDPA+GLUV3			
Metric		MSE	RMSE	MAE	MAPE	MSE	RMSE	MAE	MAPE
learning cycle	20	0.258	0.508	0.401	5.124	0.143	0.379	0.288	3.641
	30	0.351	0.593	0.507	6.238	0.156	0.395	0.311	3.943
	40	0.195	0.441	0.348	4.455	**0.097**	**0.311**	**0.236**	**3.002**
	50	0.208	0.456	0.3578	4.574	**0.097**	**0.311**	**0.252**	**3.178**
	60	0.180	0.424	0.325	4.162	**0.087**	**0.295**	**0.221**	**2.824**
	70	0.162	0.402	0.307	3.928	**0.106**	**0.326**	**0.238**	**3.043**
	80	0.156	0.395	0.300	3.822	**0.123**	**0.351**	**0.260**	**3.308**

[a]SDA:Scaled Dot-Product Attention.
[b]SDPA:Shared Double-heads Probsparse Attention.
[c]GLU:Gated Linear Unit.
[d]GLUV2:Gated Linear Unit Variants with two weight matrix matrices.
[e]GLUV3:Gated Linear Unit Variants with three weight matrix matrices.

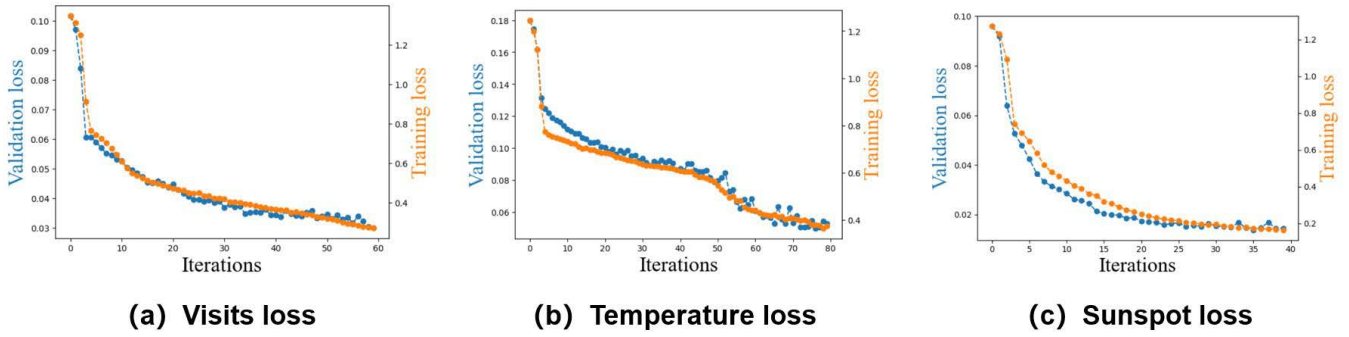


FIGURE 5. Training and validation loss.

non-large computational resources. Of course, the computing speed and results are better in a GPU environment and with specific hardware environment optimizations.

C. EXPERIMENTAL DETAILS

We split all the time series into three parts for each dataset: the training set for learning, the validation set for hyperparameter tuning, and a reserved test set for performance evaluation. We finally determine the experimental parameters, as shown in table1

In this paper, we have conducted ablation experiments, i.e., six sets of experiments, to compare the accuracy of the results obtained by our proposed components and the original components. In order to better demonstrate the accuracy of the prediction, we use mean square error (MSE), root means square error (RMSE), mean absolute error (MAE), mean absolute perceivable error (MAPE) and symmetric mean absolute percentage error (SMAPE), these five predictive evaluation indicators as benchmarks. The following shows the prediction results of related experiments we have done.

TABLE 3. Model comparison results table.

	Methods	Umformer					TFT				
	time steps	MSE	RMSE	MAE	MAPE	SMAPE	MSE	RMSE	MAE	MAPE	SMAPE
1	30	0.086	0.293	0.231	3.051	3.123	**0.061**	**0.248**	**0.198**	**2.664**	**2.686**
	60	**0.069**	**0.263**	**0.221**	**2.912**	**2.909**	0.083	0.288	0.225	2.922	2.955
	90	**0.079**	**0.281**	**0.228**	**2.914**	**2.918**	0.087	0.295	0.23	2.929	2.954
	120	**0.086**	**0.294**	**0.232**	**2.927**	**2.947**	0.109	0.331	0.269	3.327	3.365
	135	**0.086**	**0.293**	**0.227**	**2.845**	**2.869**	0.127	0.356	0.293	3.589	3.643
	185	**0.094**	**0.307**	**0.234**	**2.88**	**2.919**	0.167	0.408	0.335	4.003	4.092
		MSE	RMSE	MAE	MAPE	SMAPE	MSE	RMSE	MAE	MAPE	SMAPE
2	30	0.156	0.395	**0.305**	**4.246**	**4.318**	**0.147**	**0.383**	0.305	4.491	4.374
	60	**0.189**	**0.435**	**0.341**	**4.525**	**4.534**	0.247	0.497	0.397	5.435	5.265
	90	**0.22**	**0.469**	**0.379**	**4.782**	**4.732**	0.266	0.516	0.424	5.47	5.296
	120	**0.237**	**0.487**	**0.396**	**4.691**	**4.633**	0.324	0.569	0.469	5.657	5.473
	135	**0.247**	**0.497**	**0.399**	**4.557**	**4.497**	0.335	0.579	0.474	5.502	5.327
	185	**0.352**	**0.593**	**0.484**	**4.8**	**4.711**	0.489	0.699	0.585	5.848	5.652
		MSE	RMSE	MAE	MAPE	SMAPE	MSE	RMSE	MAE	MAPE	SMAPE
3	30	**159.094**	**12.613**	**9.658**	**10.787**	**10.948**	839.357	28.972	27.385	23.668	27.226
	60	**253.261**	**15.914**	**12.493**	**17.3**	**15.885**	581.616	24.117	21.237	20.327	23.214
	90	**286.597**	**16.929**	**13.605**	**20.021**	**20.564**	1050.375	32.409	28.57	29.701	37.269
	120	**364.196**	**19.084**	**15.567**	**22.297**	**24.088**	1595.054	39.938	34.867	34.569	44.868
	135	**382.972**	**19.57**	**16.024**	**22.002**	**23.527**	1469.998	38.341	32.818	32.21	41.554
	185	**426.72**	**20.657**	**16.85**	**20.458**	**21.629**	1151.395	33.932	27.599	26.866	33.567
		MSE	RMSE	MAE	MAPE	SMAPE	MSE	RMSE	MAE	MAPE	SMAPE
4	30	**0.002**	**0.043**	**0.035**	**0.163**	**0.163**	0.022	0.148	0.136	0.629	0.627
	60	**0.007**	**0.081**	**0.069**	**0.315**	**0.316**	0.013	0.112	0.091	0.421	0.42
	90	**0.01**	**0.102**	**0.09**	**0.413**	**0.414**	0.008	0.088	0.073	0.335	0.334
	120	**0.009**	**0.093**	**0.079**	**0.362**	**0.363**	0.009	0.095	0.074	0.342	0.342
	135	**0.008**	**0.09**	**0.076**	**0.348**	**0.349**	0.009	0.093	0.074	0.338	0.338
	185	**0.006**	**0.08**	**0.066**	**0.303**	**0.303**	0.007	0.083	0.064	0.292	0.292
		MSE	RMSE	MAE	MAPE	SMAPE	MSE	RMSE	MAE	MAPE	SMAPE
5	30	**0.211**	**0.46**	**0.404**	**4.582**	**4.493**	0.672	0.82	0.624	6.67	7.087
	60	**0.193**	**0.439**	**0.383**	**4.316**	**4.232**	0.602	0.776	0.567	6.069	6.441
	90	**0.157**	**0.396**	**0.348**	**3.852**	**3.781**	0.416	0.645	0.427	4.567	4.819
	120	**0.146**	**0.382**	**0.326**	**3.602**	**3.533**	0.444	0.667	0.445	4.748	5.016
	135	**0.143**	**0.378**	**0.325**	**3.568**	**3.5**	0.399	0.632	0.412	4.395	4.632
	185	**0.193**	**0.439**	**0.37**	**4.064**	**3.964**	0.368	0.607	0.397	4.226	4.439
	30	0.142	0.377	0.313	30.139	37.431	0.241	0.491	0.247	30.993	30.195
	60	0.157	0.396	0.327	99.506	64.407	0.275	0.524	0.341	83.103	76.343
1	90	0.14	0.375	0.312	145.465	89.605	0.14	0.375	0.312	145.465	89.605
	120	0.221	0.47	0.374	170.805	109.057	0.221	0.47	0.374	170.805	109.057
	135	0.248	0.498	0.4	188.552	114.906	0.248	0.498	0.4	188.552	114.906
	185	0.476	0.69	0.538	438.01	124.5	0.476	0.69	0.538	438.01	124.5
		MSE	RMSE	MAE	MAPE	SMAPE	MSE	RMSE	MAE	MAPE	SMAPE
2	30	0.866	0.931	0.754	10.706	11.705	6.564	2.562	2.119	31.377	26.978
	60	0.866	0.931	0.754	10.706	11.705	4.987	2.233	1.766	24.116	23.31
	90	2.769	1.664	1.49	17.662	19.806	6.819	2.611	1.999	24.673	24.978
	120	2.987	1.728	1.582	17.6	19.647	6.285	2.507	1.905	22.291	22.615
	135	3.625	1.904	1.75	18.633	20.874	6.771	2.602	1.988	22.152	22.238
	185	0.023	0.153	0.131	46.091	29.755	6.682	2.585	2.004	20.007	20.318
		MSE	RMSE	MAE	MAPE	SMAPE	MSE	RMSE	MAE	MAPE	SMAPE
3	30	178.996	13.379	11.811	13.727	12.596	1665.928	40.816	34.092	42.053	34.444
	60	581.424	24.113	20.425	31.448	24.859	1818.58	42.645	35.171	45.602	42.469
	90	3513.537	59.275	44.399	96.211	50.068	2673.534	51.706	42.124	51.903	62.158
	120	5500.447	74.165	58.096	118.099	59.018	3166.385	56.271	46.122	73.177	74.322
	135	4982.304	70.585	54.091	107.514	54.58	3243.22	56.949	47.106	73.239	71.563
	185	4020.305	63.406	48.986	86.768	49.871	3366.006	58.017	48.244	73.147	65.722
		MSE	RMSE	MAE	MAPE	SMAPE	MSE	RMSE	MAE	MAPE	SMAPE
4	30	0.177	0.421	0.331	1.519	1.525	0.005	0.07	0.06	0.274	0.274
	60	0.199	0.446	0.362	1.662	1.654	0.05	0.223	0.169	0.775	0.78
	90	0.283	0.532	0.397	1.822	1.802	0.145	0.381	0.305	1.399	1.414
	120	0.322	0.568	0.447	2.048	2.022	0.2	0.447	0.378	1.73	1.751
	135	0.314	0.561	0.445	2.038	2.013	0.188	0.434	0.366	1.677	1.697
	185	0.29	0.538	0.437	2.003	1.978	0.265	0.515	0.442	2.024	2.025

TABLE 3. Model comparison results table.

		MSE	RMSE	MAE	MAPE	SMAPE	MSE	RMSE	MAE	MAPE	SMAPE
5	30	0.733	0.856	0.663	7.056	7.506	0.626	0.791	0.589	6.326	6.717
	60	0.661	0.813	0.601	6.403	6.808	0.557	0.747	0.537	5.779	6.125
	90	0.457	0.676	0.453	4.814	5.089	0.386	0.621	0.406	4.355	4.588
	120	0.486	0.697	0.47	4.987	5.279	0.41	0.64	0.425	4.555	4.803
	135	0.435	0.66	0.432	4.582	4.841	0.37	0.608	0.398	4.264	4.481
	185	0.402	0.634	0.412	4.368	4.601	0.341	0.584	0.387	4.144	4.337
1	30	0.122	0.35	0.281	3.984	3.873	0.115	0.339	0.27	3.83	3.727
	60	0.354	0.595	0.489	6.96	6.626	0.327	0.572	0.467	6.626	6.32
	90	0.872	0.934	0.752	10.727	9.925	0.813	0.901	0.721	10.223	9.483
	120	1.028	1.014	0.85	12.144	11.195	0.945	0.972	0.809	11.484	10.622
	135	1.11	1.053	0.889	12.713	11.689	1.015	1.007	0.844	11.975	11.051
	185	1.458	1.208	0.992	14.219	12.91	1.323	1.15	0.932	13.225	12.059
		MSE	RMSE	MAE	MAPE	SMAPE	MSE	RMSE	MAE	MAPE	SMAPE
2	30	4.415	2.101	1.78	22.488	23.337	4.575	2.139	1.815	23.021	23.774
	60	3.244	1.801	1.471	18.571	18.993	3.346	1.829	1.487	18.923	19.199
	90	5.462	2.337	1.777	22.428	22.626	5.637	2.374	1.794	23.013	22.848
	120	7.07	2.659	2.041	25.747	24.482	7.525	2.743	2.093	27.042	25.137
	135	9.744	3.122	2.381	30.038	26.981	10.566	3.251	2.467	32.043	28.036
	185	17.81	4.22	3.251	41.01	33.095	19.83	4.453	3.435	45.247	35.316
		MSE	RMSE	MAE	MAPE	SMAPE	MSE	RMSE	MAE	MAPE	SMAPE
3	30	172.114	13.119	10.692	10.415	10.169	171.329	13.089	10.64	10.377	10.118
	60	1088.657	32.995	26.173	25.491	31.839	1084.682	32.935	26.103	25.447	31.775
	90	2770.535	52.636	43.126	42.003	62.498	2763.466	52.569	43.051	41.966	62.439
	120	3439.913	58.651	49.329	48.043	74.473	3431.689	58.581	49.251	48.009	74.416
	135	3302.341	57.466	48.341	47.081	69.831	3295.676	57.408	48.278	47.06	69.785
	185	2942.983	54.249	45.642	44.452	60.272	2939.017	54.213	45.604	44.453	60.247
		MSE	RMSE	MAE	MAPE	SMAPE	MSE	RMSE	MAE	MAPE	SMAPE
4	30	0.013	0.112	0.104	0.477	0.476	0.011	0.103	0.095	0.437	0.436
	60	0.011	0.106	0.095	0.437	0.437	0.012	0.111	0.097	0.449	0.449
	90	0.026	0.16	0.14	0.648	0.65	0.036	0.189	0.16	0.74	0.743
	120	0.033	0.182	0.16	0.745	0.747	0.049	0.222	0.192	0.889	0.893
	135	0.033	0.183	0.16	0.745	0.747	0.046	0.215	0.185	0.856	0.86
	185	0.322	0.567	0.394	1.836	1.806	0.286	0.535	0.388	1.805	1.781
		MSE	RMSE	MAE	MAPE	SMAPE	MSE	RMSE	MAE	MAPE	SMAPE
5	30	0.215	0.463	0.373	4.156	4.048	0.206	0.454	0.355	3.982	3.868
	60	0.549	0.741	0.595	6.486	6.565	0.534	0.731	0.578	6.305	6.389
	90	0.47	0.686	0.574	6.236	6.231	0.455	0.674	0.558	6.059	6.059
	120	0.477	0.691	0.585	6.342	6.37	0.466	0.683	0.572	6.196	6.231
	135	0.492	0.701	0.603	6.532	6.52	0.479	0.692	0.59	6.384	6.379
	185	0.507	0.712	0.627	6.781	6.727	0.494	0.703	0.615	6.643	6.598

[a] 1 is the Visits dataset
[b] 2 is the Temperature dataset
[c] 3 is the Sunspot dataset
[d] 4 is the ECN dataset
[e] 5 is the Taxi dataset

In table 2, we divided the training counts into seven categories and respectively trained [20,30,40,50,60,70,80] times on the training set and tested them on the test set. The final prediction results are compared for different components and different training times.The experimental results demonstrate that the SDHPA and GLUV3 proposed in this paper have a better performance compared to the traditional attention mechanism and gating unit.

The figure5 shows the loss of Umformer model training and validation in each dataset.

1) BASELINES

Based on the three data sets above, we chose five-time series forecasting methods for comparison, including TFT, DeepAR, Prophet, ARMA, and ARIMA, to better demonstrate the model's performance in multi-step time series forecasting in univariate data sets. We calculate the result set of each model prediction in sections and calculate the average error of 30, 60, 90, 120, 135, and 185 time steps respectively. In order to better show the accuracy of the prediction, we use MSE, RMSE, MAE, MAPE and SMAPE as benchmarks. The experimental results are shown in Table 3. The results demonstrate the advantages of umformer on UMTF problems.

2) FEATURE SELECTION

For univariate datasets, feature selection often needs to be performed after univariate feature decomposition, and

the methods used are generally different for the various datasets. We add to different datasets their large number of relevant attributes that affect the final prediction results. And in the prophet-based feature decomposition method mentioned above, the output features are to some extent unknown in terms of their impact on the prediction results. We need to extract the relevant features that are useful for prediction, and discard irrelevant and redundant features. This will help to improve model accuracy, reduce algorithm learning time and also increase the interpretability of the model.

In the experimental process, we mainly used a Decision Tree(DT) based feature selection method, using information gain as the evaluation function. For example, The C4.5 algorithm is used in the training set until the DT grows sufficiently, and then pruning is performed using the evaluation function. Finally, the concatenation of all feature subsets appearing on the path of any one leaf node is the result of feature selection.

The DT lists all feasible solutions to the decision problem and the various possible states of nature, as well as the expected values of each feasible method in the various states, and provides a visual representation of the entire decision problem at different stages of the decision process in time and decision sequence. Figure 7,Figure 8 shows a graph of the weights of each feature after the model features have been selected. To improve the ease of operation and generalisation of the model, we have embedded the feature selection algorithm into the model. This includes algorithms such as ID3, C4.5, CART, etc. To ensure the scientific nature of the experimental results, we uniformly used random forest for feature selection in our comparison experiments. It is worth noting that other algorithms can also achieve satisfactory results in practical applications.

3) HYPERPARAMETER OPTIMISATION ALGORITHMS

All the research described above in this paper is an optimisation of the algorithm itself. In addition, finding the optimal hyperparameters is also an essential part of the research. Taking different values of the hyperparameters has different effects on the performance of the model, and grid search and stochastic search are important ways of optimising the hyperparameters. Finding the optimal hyperparameters from the hyperparameter space, grid search can be seen almost as violently trying to select the most optimal set of hyperparameters from each set of parameters in the parameter space, which is obviously not efficient. Stochastic search has been shown in our experiments to be a more efficient method, which allows the computational cost to be chosen independently of the number of parameters and possible values, and adding parameters that do not affect performance does not reduce efficiency. To fulfil the experimental requirements of this study, we chose to use stochastic search for hyperparameter optimisation. Previous research has provided us with a great deal of assistance [66].

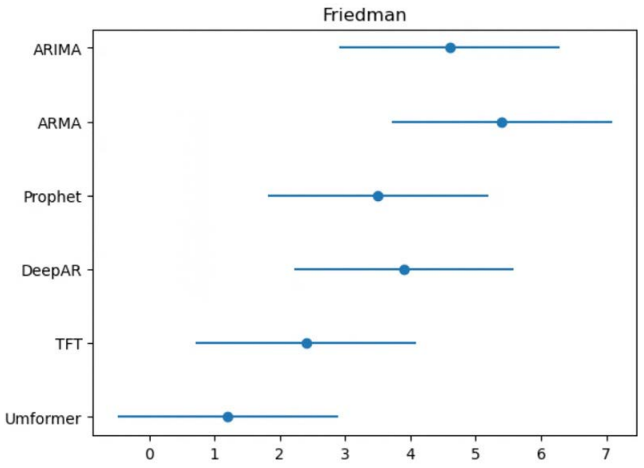


FIGURE 6. Friedman and nemenyi test.

4) FRIEDMAN AND NEMENYI TEST

In order to compare the generalisation performance of different learning algorithms across the board, it is not enough to rely on a measure of sexiness on a particular dataset. We need to use hypothesis testing, which provides an important basis for our algorithm comparisons. Also we generally need to compare the performance of multiple algorithms on multiple datasets, and here the Friedman test and the Nemenyi test are often used for comparison.

We tested the MSE of the results of each model in 185 long series predictions. The hypothesis test rejected the hypothesis that the performance of the six algorithms did not differ across the five data sets. This indicates that the algorithms perform significantly differently, at which point a follow-up test is required to further distinguish each algorithm. We calculated the critical value domain CD $= 3.372$ for the difference in mean ordinal values by the Nemenyi test and the Friedman plot is shown figure 6. It is demonstrated that the algorithms differ significantly directly and that umformer has a greater advantage over the other algorithms.

5) ANALYSIS OF CHARACTERISTIC VARIABLES

During the experiment, we specifically analyzed the influence of various variables on the experiment. We analyze all the features extracted in feature engineering and examine the weights of variables affecting predictions. It enables us to deeply analyze the influence of various variables on the prediction results, and to more flexibly select various variables to input into the model when solving practical problems.

As shown in the figure 7 and 8, the influence degree of each variable on the prediction is displayed. We use the visualization tool to show the influence weight of different variables on the prediction result. This is the input after we remove some variables with small influence. For different datasets, our input variables to the model are different. The results show that known future variables have an impact on the predicted results, especially some cyclical variables, it is necessary to input into the model.

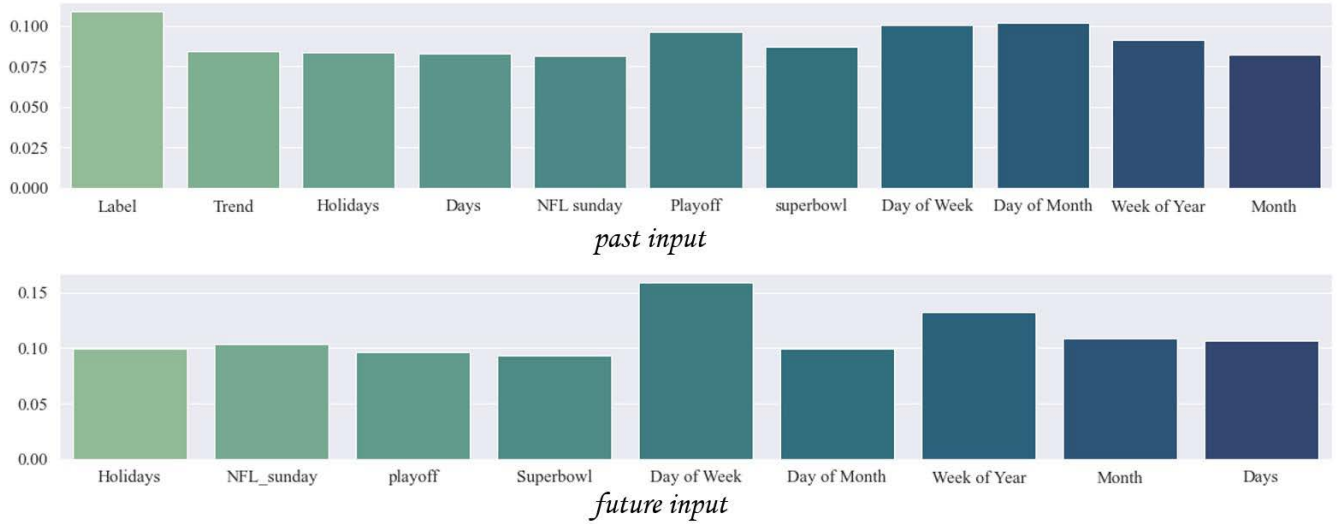


FIGURE 7. Weights of visits dataset variables.

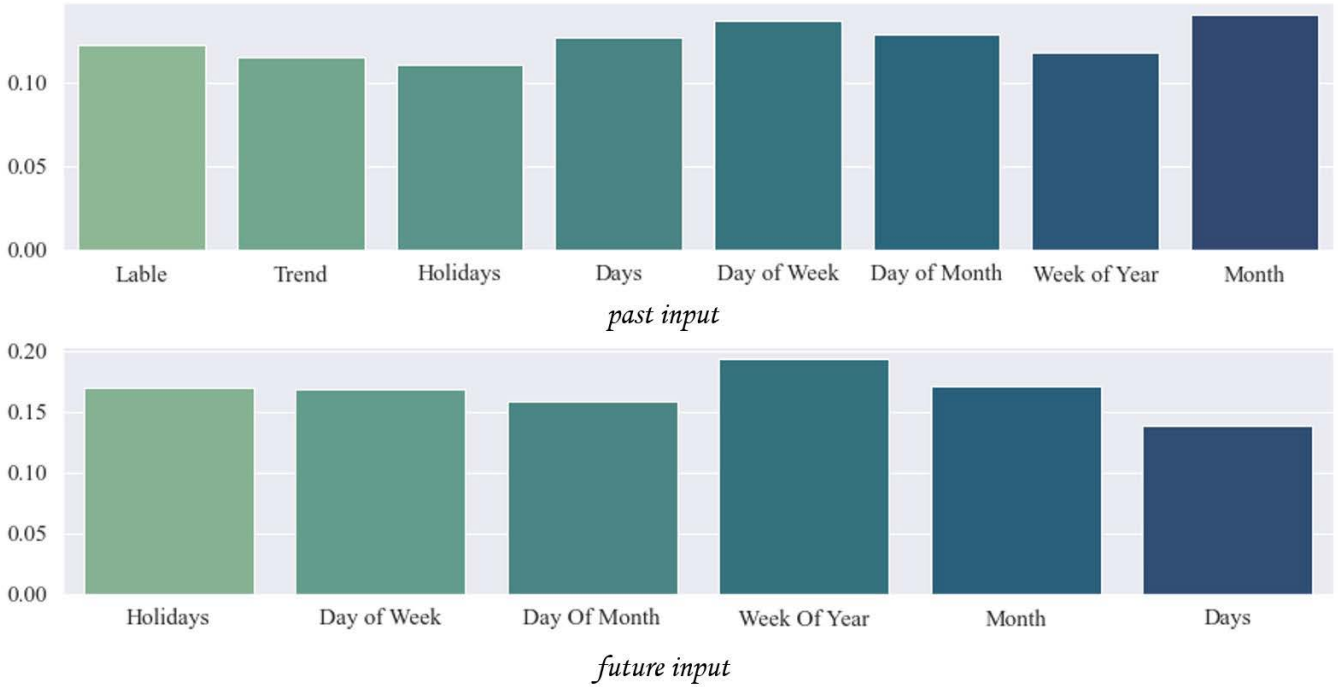


FIGURE 8. Weights of temperature dataset variables.

For the three data sets, we entered 11, 8, and 8 related variables respectively. It can be seen that the effect of periodic terms such as weeks and months is higher than that of other variables. For datasets with strong periodicity, it is necessary for us to enter the periodic term when entering.

D. RESULTS ANALYSIS

Table 2 shows a comparison between our improved component and the original component. We show the results of the impact of different epochs through experiments comparing the accuracy of different components on the same dataset. It is finally demonstrated that SDHPA and GLUV3 as an improved Attention mechanism and gated linear units result in more significant prediction accuracy after the model is trained to a certain level. It is worth noting that Scaled Dot-Product Attention+GLUV2 may also be helpful at certain times that require further research in the future. SDHPA and GLUV3 in the Umformer model are excellent structures to use for prediction.

Table 3 summarizes the prediction evaluation results for the three datasets across the six methods. As the demand for predictive power increases, we gradually lengthen the prediction time step. The best results are highlighted in bold. Compared with some existing models, the results of

umformer for univariate prediction are satisfactory. When solving specific problems in life, we need to consider the impact of correlated variables on predictions. Most univariate data have strong periodicity, which makes us need to consider various cyclical variables, as well as various continuous variables, when making predictions. Furthermore, dividing past variables and future variables is a very efficient approach for model training. When the model predicts, it is necessary for us to input known time variables, as shown in Figure 2, which can greatly improve model performance.

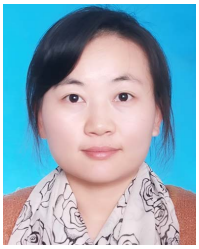
VI. CONCLUSION

This paper studies the univariate multi-step time series prediction and proposes a Umformer Framework. Specifically, we propose a temporal feature extraction approach based on the Prophet algorithm to decompose the univariate time series, design a SDHPA to deal with the complexity, interpretability, and accuracy problems in the Attention mechanism in Transformer, and design GLUV3 as a novel network layer to play the role of feature information extraction in the model. We demonstrate the optimism of the method's predictive power in univariate temporal prediction problems with several publicly available datasets.

REFERENCES

[1] T. Ferenti, "Biomedical applications of time series analysis," in *Proc. IEEE 30th Neumann Colloq. (NC)*, Nov. 2017, pp. 000083–000084.
[2] M. Abe and H. Nakayama, "Deep learning for forecasting stock returns in the cross-section," in *Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining*. Springer, 2018, pp. 273–284.
[3] L. Cai, K. Janowicz, G. Mai, B. Yan, and R. Zhu, "Traffic transformer: Capturing the continuity and periodicity of time series for traffic forecasting," *Trans. GIS*, vol. 24, no. 3, pp. 736–755, Jun. 2020.
[4] R. Chuentawat and Y. Kan-ngan, "The comparison of PM2.5 forecasting methods in the form of multivariate and univariate time series based on support vector machine and genetic algorithm," in *Proc. 15th Int. Conf. Electr. Eng./Electron., Comput., Telecommun. Inf. Technol. (ECTI-CON)*, Jul. 2018, pp. 572–575.
[5] Y.-Y. Chang, F.-Y. Sun, Y.-H. Wu, and S.-D. Lin, "A memory-network based solution for multivariate time-series forecasting," 2018, *arXiv:1809.02105*.
[6] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long- and short-term temporal patterns with deep neural networks," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jun. 2018, pp. 95–104.
[7] S. J. Taylor and B. Letham, "Forecasting at scale," *Amer. Statistician*, vol. 72, no. 1, pp. 37–45, 2018.
[8] B. Lim, S. Ö. Arik, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *Int. J. Forecasting*, vol. 37, no. 4, pp. 1748–1764, Oct. 2021.
[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, U. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[10] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 933–941.
[11] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling," in *Proc. Interspeech*, Sep. 2012, pp. 1–4.
[12] S. D. Balkin and J. K. Ord, "Automatic neural network modeling for univariate time series," *Int. J. Forecasting*, vol. 16, no. 4, pp. 509–515, Oct. 2000.
[13] J. du Preez and S. F. Witt, "Univariate versus multivariate time series forecasting: An application to international tourism demand," *Int. J. Forecasting*, vol. 19, no. 3, pp. 435–451, Jul. 2003.
[14] J. C. Cuaresma, J. Hlouskova, S. Kossmeier, and M. Obersteiner, "Forecasting electricity spot-prices using linear univariate time-series models," *Appl. Energy*, vol. 77, no. 1, pp. 87–106, 2004.
[15] G. Papacharalampous, H. Tyralis, and D. Koutsoyiannis, "Univariate time series forecasting of temperature and precipitation with a focus on machine learning algorithms: A multiple-case study from Greece," *Water Resour. Manage.*, vol. 32, no. 15, pp. 5207–5239, Dec. 2018.
[16] O. Yazdanbakhsh and S. Dick, "Multi-variate timeseries forecasting using complex fuzzy logic," in *Proc. Annu. Conf. North Amer. Fuzzy Inf. Process. Soc. (NAFIPS) Held Jointly 5th World Conf. Soft Comput. (WConSC)*, Aug. 2015, pp. 1–6.
[17] S. S. Ratakonda and S. Sasi, "Seasonal trend analysis on multi-variate time series data," in *Proc. Int. Conf. Data Sci. Eng. (ICDSE)*, Aug. 2018, pp. 1–6.
[18] S. Liu, S. R. Poccia, K. S. Candan, M. L. Sapino, and X. Wang, "Robust multi-variate temporal features of multi-variate time series," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 14, no. 1, pp. 1–24, Jan. 2018.
[19] D. A. Dickey and W. A. Fuller, "Distribution of the estimators for autoregressive time series with a unit root," *J. Amer. Statist. Assoc.*, vol. 74, no. 366a, pp. 427–431, 1979.
[20] W. A. Fuller and D. P. Hasza, "Properties of predictors for autoregressive time series," *J. Amer. Stat. Assoc.*, vol. 76, no. 373, pp. 155–161, Mar. 1981.
[21] V. Haggan and T. Ozaki, "Modelling nonlinear random vibrations using an amplitude-dependent autoregressive time series model," *Biometrika*, vol. 68, no. 1, pp. 189–196, 1981.
[22] J. M. Lucas and M. S. Saccucci, "Exponentially weighted moving average control schemes: Properties and enhancements," *Technometrics*, vol. 32, no. 1, pp. 1–12, 1990.
[23] S. Hansun, "A new approach of moving average method in time series analysis," in *Proc. Conf. New Media Stud. (CoNMedia)*, Nov. 2013, pp. 1–4.
[24] S. Makridakis, S. C. Wheelwright, and R. J. Hyndman, *Forecasting Methods and Applications*. Hoboken, NJ, USA: Wiley, 2008.
[25] J. L. Torres, A. García, M. De Blas, and A. De Francisco, "Forecast of hourly average wind speed with ARMA models in Navarre (Spain)," *Sol Energy*, vol. 79, no. 1, pp. 65–77, 2005.
[26] S.-J. Huang and K.-R. Shih, "Short-term load forecasting via ARMA model identification including non-Gaussian process considerations," *IEEE Trans. Power Syst.*, vol. 18, no. 2, pp. 673–679, May 2003.
[27] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 5243–5253.
[28] J. Contreras, R. Espinola, F. J. Nogales, and A. J. Conejo, "ARIMA models to predict next-day electricity prices," *IEEE Trans. Power Syst.*, vol. 18, no. 3, pp. 1014–1020, Aug. 2003.
[29] S. Ho and M. Xie, "The use of ARIMA models for reliability forecasting and analysis," *Comput. Ind. Eng.*, vol. 35, nos. 1–2, pp. 213–216, 1998.
[30] D. Benvenuto, M. Giovanetti, L. Vassallo, S. Angeletti, and M. Ciccozzi, "Application of the ARIMA model on the COVID-2019 epidemic dataset," *Data Brief*, vol. 29, Apr. 2020, Art. no. 105340.
[31] E. S. Gardner, Jr., "Exponential smoothing: The state of the art," *J. Forecasting*, vol. 4, no. 1, pp. 1–28, 1985.
[32] R. Hyndman, A. B. Koehler, J. K. Ord, and R. D. Snyder, *Forecasting With Exponential Smoothing: The State Space Approach*. Springer, 2008.
[33] E. Ostertagová and O. Ostertag, "The simple exponential smoothing model," in *Proc. 4th Int. Conf. Modelling Mech. Mechatronic Syst., Tech. Univ. Košice, Slovak Republic*, 2011, pp. 380–384.
[34] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on LSTM recurrent neural network," *IEEE Trans. Smart Grid*, vol. 10, no. 1, pp. 841–851, Jan. 2019.
[35] Z. Zhao, W. Chen, X. Wu, P. C. Chen, and J. Liu, "LSTM network: A deep learning approach for short-term traffic forecast," *IET Intell. Transp. Syst.*, vol. 11, no. 2, pp. 68–75, 2017.
[36] X. Qing and Y. Niu, "Hourly day-ahead solar irradiance prediction using weather forecasts by LSTM," *Energy*, vol. 148, pp. 461–468, Apr. 2018.
[37] S. Siami-Namini, N. Tavakoli, and A. Siami Namin, "A comparison of ARIMA and LSTM in forecasting time series," in *Proc. 17th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2018, pp. 1394–1401.
[38] N. Wu, B. Green, X. Ben, and S. O'Banion, "Deep transformer models for time series forecasting: The influenza prevalence case," 2020, *arXiv:2001.08317*.
[39] M. Xu, W. Dai, C. Liu, X. Gao, W. Lin, G.-J. Qi, and H. Xiong, "Spatial-temporal transformer networks for traffic flow forecasting," 2020, *arXiv:2001.02908*.

[40] S. Wang, L. Zhou, Z. Gan, Y.-C. Chen, Y. Fang, S. Sun, Y. Cheng, and J. Liu, "Cluster-former: Clustering-based sparse transformer for long-range dependency encoding," 2020, *arXiv:2009.06097*.
[41] J. Rebane, I. Karlsson, P. Papapetrou, and S. Denic, "Seq2seq RNNs and ARIMA models for cryptocurrency prediction: A comparative study," in *Proc. SIGKDD Fintech*, London, U.K., Aug. 2018.
[42] Y. Zhang, Y. Li, and G. Zhang, "Short-term wind power forecasting approach based on Seq2Seq model using NWP data," *Energy*, vol. 213, Dec. 2020, Art. no. 118371.
[43] P. Lara-Benítez, L. Gallego-Ledesma, M. Carranza-García, and J. M. Luna-Romera, "Evaluation of the transformer architecture for univariate time series forecasting," in *Proc. Conf. Spanish Assoc. Artif. Intell.* Springer, 2021, pp. 106–115.
[44] C. Aicher, N. J. Foti, and E. B. Fox, "Adaptively truncating backpropagation through time to control gradient bias," in *Uncertainty in Artificial Intelligence*. PMLR, 2020, pp. 799–808.
[45] T. Trinh, A. Dai, T. Luong, and Q. Le, "Learning longer-term dependencies in rnns with auxiliary losses," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 4965–4974.
[46] D. Lukovnikov and A. Fischer, "Improving breadth-wise backpropagation in graph neural networks helps learning long-range dependencies," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 7180–7191.
[47] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," 2017, *arXiv:1707.01926*.
[48] K. Qi, H. Yang, C. Li, Z. Liu, M. Wang, Q. Liu, and S. Wang, "X-net: Brain stroke lesion segmentation based on depthwise separable convolution and long-range dependencies," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* Springer, 2019, pp. 247–255.
[49] S. Khorram, Z. Aldeneh, D. Dimitriadis, M. McInnis, and E. Mower Provost, "Capturing long-term temporal dependencies with convolutional networks for continuous emotion recognition," 2017, *arXiv:1708.07050*.
[50] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," 2015, *arXiv:1506.07503*.
[51] J. Qiu, H. Ma, O. Levy, S. Wen-tau Yih, S. Wang, and J. Tang, "Blockwise self-attention for long document understanding," 2019, *arXiv:1911.02972*.
[52] N. Kitaev, Ł. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," 2020, *arXiv:2001.04451*.
[53] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," 2019, *arXiv:1901.02860*.
[54] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proc. AAAI*, 2021, pp. 1–10.
[55] H. Song, D. Rajan, J. J. Thiagarajan, and A. Spanias, "Attend and diagnose: Clinical time series analysis using attention models," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–8.
[56] J. Ma, Z. Shou, A. Zareian, H. Mansour, A. Vetro, and S.-F. Chang, "CDSA: Cross-dimensional self-attention for multivariate, geo-tagged time series imputation," 2019, *arXiv:1905.09904*.
[57] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," 2020, *arXiv:2004.05150*.
[58] H. Akaike, "Autoregressive model fitting for control," in *Selected Papers of Hirotugu Akaike*. Springer, 1998, pp. 153–170.
[59] G. C. Tiao and M. R. Grupe, "Hidden periodic autoregressiye-moving average models in time series data," *Biometrika*, vol. 67, no. 2, pp. 365–373, 1980.
[60] R. L. Kashyap, "Optimal choice of AR and MA parts in autoregressive moving average models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-4, no. 2, pp. 99–104, Mar. 1982.
[61] W. Lu, J. Li, Y. Li, A. Sun, and J. Wang, "A CNN-LSTM-based model to forecast stock prices," *Complexity*, vol. 2020, Nov. 2020, Art. no. 6622927.
[62] G. Gong, X. An, N. K. Mahato, S. Sun, S. Chen, and Y. Wen, "Research on short-term load prediction based on seq2seq model," *Energies*, vol. 12, no. 16, p. 3199, Aug. 2019.
[63] A. Mnih and G. Hinton, "Three new graphical models for statistical language modelling," in *Proc. 24th Int. Conf. Mach. Learn. (ICML)*, 2007, pp. 641–648.
[64] N. Shazeer, "GLU variants improve transformer," 2020, *arXiv:2002.05202*.
[65] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," 2016, *arXiv:1606.08415*.
[66] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2011, pp. 1–9.



MIN LI received the master's degree in communications engineering from Tianjin University. She was a Professor in software engineering with the Qilu University of Technology (Shandong Academic of Sciences). Her research interests include information technology standardization, economic and information development, big data analysis and application, data governance and data openness, and digital government planning and evaluation.

QINGHUI CHEN received the bachelor's degree in electronic information science and technology. He is currently pursuing the master's degree with the Department of Computing, Qilu University of Technology (Shandong Academic of Sciences). His research interests include time series forecasting, machine learning, and deep learning.

GANG LI received the master's degree in management from Shandong University and the Ph.D. degree in management science and engineering from the Harbin Institute of Technology. He was a Professor in software engineering with the Qilu University of Technology (Shandong Academic of Sciences). His research interests include big data analytics applications, data governance, digital economy, and digital government.

DELONG HAN received the Ph.D. degree in electronic science and technology from the Beijing University of Posts and Telecommunications. He is currently a Research Associate with the Qilu University of Technology (Shandong Academic of Sciences). His research interests include digital government, open government data, and digital economy.